7E−4

面積最小化を目的とする多段論理合成用状態割り付け手法

横山 昌生、　野田 浩明、　高橋 瑞樹、　神戸 尚志

シャープ(株)

1　はじめに

集積回路技術の進展に伴ってVLSI化可能な回路規模が増大する反面、その設計に要するコストもまた莫大になりつつあり、大規模回路を低コストで設計し得る技術が求められている。この問題に対処すべく、我々はこれまでにレジスタ転送レベルからの自動合成システム[1]を開発した。

本自動合成システムでは、制御回路を一つの同期式順序回路とみなして合成する。本稿では、その処理中で用いている状態割り付け手法について述べる。

2　状態割り付け

同期式順序回路の動作仕様は、例えば図1のような状態遷移表（この例では、入力数1、出力数2、状態数4）として与えられる。そこで、状態割り付け、即ち記号表現された状態各々にコードを割り当てる操作を行う。各状態コードと状態遷移表から、次状態コード及び外部出力を表す論理関数（本稿では、次状態関数及び外部出力関数と呼ぶ）が得られ、それを論理合成して組合せ回路として実現する。この組合せ回路と、状態コード保持用の同期レジスタを組み合わせたものが、求める同期式順序回路となる。

さて、得られる順序回路が動作仕様を満たしさえすればよいのであれば、コードから状態が一意に定まる限り、状態各々にどんなコードを割り付けるのも自由である。しかし、状態とコードの対応を変えれば、得られる論理関数、延いては論理合成後の組合せ回路も変わる。よって、論理合成だけでなく、状態割り付けを工夫することで、より回路の最適化を図ることができる。

2.1　目的

本稿で述べる状態割り付け手法は、割り付け後に多段論理合成を行うことを前提とした上で、同期式順序回路の面積最小化を目指したものである。状態コード長、即ちレジスタのビット長は最短（$\lceil \log_2($状態数$) \rceil$）とする。そして、次状態関数に含まれるリテラル数が、論理合成によってより多く削減される様に工夫することで面積最小化を目指す。

入力 X	現状態	次状態	出力
0	S_0	S_1	0 0
1	S_0	S_2	0 1
0	S_1	S_1	0 1
1	S_1	S_0	0 0
0	S_2	S_0	1 0
1	S_2	S_3	1 0
0	S_3	S_0	1 0
1	S_3	S_3	0 0

図1．状態遷移表の例

2.2　次状態関数

図1で、現状態が S_i であることを表す項を Y_i、次状態が S_j になることを表す論理式を Z_j とおくと、

$$Z_0 = XY_1 + \overline{X}Y_2 + \overline{X}Y_3$$
$$Z_1 = \overline{X}Y_0 + \overline{X}Y_1$$
$$Z_2 = XY_0$$
$$Z_3 = XY_2 + XY_3$$

と表すことができる。次状態関数は、状態コードの各桁の値に基いて、いくつかのZ式の和となる。例えば、S_0, S_1, S_2, S_3 に、00,01,11,10 を割り付ければ、状態コードの1桁目に対応する次状態関数は $Z_2 + Z_3$、2桁目に対応するそれは $Z_1 + Z_2$ となる。

2.3　従来手法

同期式順序回路の面積最小化を目標とする手法としては、mustang[2]が広く知られており、これまでの研究例を見てもmustangを基にしたものが数多く見られる。mustangでは、fan-out指向アルゴリズム（外部出力関数の簡単化に効果がある）とfan-in指向アルゴリズム（次状態関数の簡単化に効果がある）の2つの手法が提案されている。

fan-in指向アルゴリズムでは、Z式の全てのペアについて、共通の入力リテラル、共通の現状態項を多く持つ場合に、対応する2状態にハミング距離の近いコードを割り付ける。その効果を考えて見ると、例えば Z_0 と Z_3 では Y_2, Y_3 を共通項に持つので、S_0, S_3 に近いコードとして11,01を割り付けたとすると、コードの2桁目に対応する次状態関数は $Z_0 + Z_3$ となり、

Area-Optimal State Assignment for Multi-Level Logic Synthesis

Masao Yokoyama, Hiroaki Noda, Mizuki Takahashi and Takashi Kambe

SHARP Corporation

そこから Y_2, Y_3 が括り出せて、次状態関数に含まれるリテラル数は減る。

2.4 従来手法の問題点

共通項に着目し、それを括り出すことでリテラル数は減らせる。しかし、括り出せる部分が多くても、項数やリテラル数を多く含む式を積極的に選択していると、効果は相殺されてしまう場合も多々ある。

また先の例では、S_0,S_3 に 11,01 を割り付けた結果次状態関数に Z_0+Z_3 が現れたが、仮に（コード間の距離は同じでも）00,01 を割り付けていれば現れない。2つの状態コードの同じ桁が共に1になることによって、対応する2つのZ式が1つの次状態関数中に現れるのだから、コード間の距離に着目しただけでは、意図する2式が何個の次状態関数中に現れるのかを操作できない。

2.5 本手法

Z_i と Z_j の和をとる時、リテラル数削減を考えるならば、共通項を多く取り得るかどうかだけでなく、そもそも各式に含まれているリテラル数も考え合わせる必要が認められる。本手法では、Z_i,Z_j に対する評価関数 f_{ij} を、

Z_i に含まれるリテラル数 +
Z_j に含まれるリテラル数 +
Z_i,Z_j から括り出して減らせる
リテラル数（の期待値）

とした。次状態関数を、f_{ij} がより小さい Z_i,Z_j の和として構成できれば、リテラル数削減が図れる。

そこで次に、f_{ij} がより小さいような S_i,S_j の状態コードの同じ桁が共に1になるように状態コードを決めて行く。これで狙いどうり、次状態関数を f_{ij} がより小さい Z_i,Z_j の和として構成でき、リテラル数は削減される。

具体的には、以下の手順でコードを決めて行く。

- 全状態集合を、コードの1桁目に1を割り付ける集合（f_{ij} がより小さいような S_i,S_j からなる）と0を割り付ける集合に2分割する。
- 各々の集合を源集合とし、f_{ij} に基き、コードの次の桁に1を割り付ける集合と0を割り付ける集合に2分割する。この処理を再帰的に繰り返す。

この分割に従って、1桁目、2桁目、...とコードを決定する。

3 評価実験

MCNCベンチマークデータによる実験結果を表1に示す。比較のための割り付け手法には、ランダム割り付けと mustang の fan-in 指向アルゴリズムを用いた（コード長は何れも最短とした）。表の各欄は、各々の手法で状態割り付けし、clover[3] で次状態関数と外部出力関数を論理合成した後の、factored form でのリテラル数を表している（ランダム割り付けに関しては、割り付けと合成を5回試み、その平均を取った）。この表では、本手法によって平均でランダム割り付けより22%、mustang より7%程の面積削減がなされたことが示されている。なおCPU時間としては、ここに挙げた数十ゲート規模の回路の状態割り付けに、1秒弱（15 MIPS Work-station 上）要した。

表1．実験結果

データ	入力数	出力数	状態数	ランダム	mustang	本手法
bbara	4	2	10	120	89	71
bbsse	7	7	16	185	165	151
dk14	3	5	7	156	143	126
ex4	6	9	14	113	102	101
ex6	5	8	8	145	124	128
mark1	5	16	15	142	137	129
opus	5	6	10	152	114	107
s8	4	1	5	79	69	66
sse	7	7	16	192	165	151

4 おわりに

本状態割り付け手法は、

1. 評価関数で、（共通項の括り出し数だけでなく、）リテラル数を包括的に見積もる。
2. コード決定では、（コード間の距離のみ考えるのではなく、）桁毎の値を揃える操作によって、意図する2式が同じ次状態関数に含まれるようにする。

という特徴を持ち、同期式順序回路の面積削減に効果がある。

現版では、外部出力関数のリテラル数削減を考慮しておらず、この点が検討課題と言える。また、評価関数（例えば、複数（3以上）状態間の関係も見積もる）やコード決定法（例えば、コード決定に関与した評価関数値の総和の最小化を図る）についても、考慮の余地がある。

参考文献

[1] 野田、高橋、横山、神戸：“レジスタ転送レベル合成システム(1)”，1991年電子情報通信学会春季全国大会.

[2] S.Devadas,et al.：“MUSTANG: State Assignment of Finite State Machines Targeting Multi-Level Logic Implementation” IEEE Trans. on CAD, vol.7, Dec. 1988.

[3] 三浦、竹田、神戸：“多段論理合成における時間最適化の一手法”，信学技報，VLD89-85(1989).